**=====================OPENCV BUILD=====================**

For Windows

Ver 0.0.1

Hi! (Sorry for my bad english)

In this tutorial, we’ll build an opencv library from source.

**Tested on my laptop with this specification:**

1. Intel i5-5200U (Broadwell)
2. GPU:
   1. Intel HD Graphics 5500
   2. Nvidia GeForce 940M (Compute Capability 5.0) (Honestly, I haven’t tested on AMD GPU; If you have AMD GPU, try with OpenCL instead CUDA. CUDA based on Nvidia arch)
3. 1 TB HDD
4. Operating System: Windows 10 Pro (64-bit) (I guess different version its doesn’t matter. But I’ve tested it on Ubuntu 16.10 (64-bit) and I got some trouble (It was my fault; make sure to install CUDA properly). If you had more experience with Linux you can choose it, but if you haven’t, I suggested to use windows instead)
5. Another peripheral: integrated webcam

**Dependencies:**

\*Note: Below here, I wrote everything I did for windows. If you use Linux, you can install some of this dependencies and use ‘make’ instead visual studio. You can still install CUDA and cuDNN on Linux.

1. Python 3.5 🡪 Better use conda and change it to 3.5 (I’ve tried without conda and found it hard because there’s several library that you had to install). [Link](https://anaconda.org/) (Anaconda 3)

* If you use python without conda, make sure to install numpy and matplotlib ([Link](http://www.lfd.uci.edu/~gohlke/pythonlibs/))

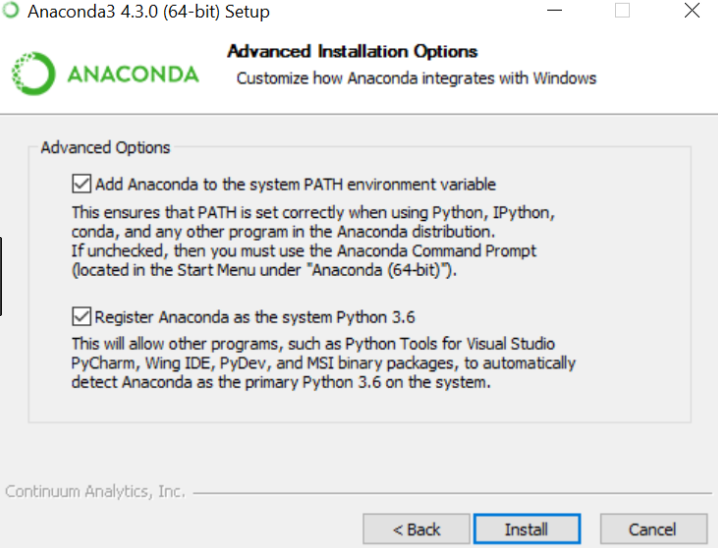
1. Visual studio 2015 (I used update 3 version. [Link](https://www.visualstudio.com/vs/older-downloads/)). CAUTIONS: Before installing visual studio, you had to install Visual C++ Redistibutable Packages ([link](https://www.microsoft.com/en-us/download/details.aspx?id=48145)). I found that Visual Studio 2015 has no C++ library. After that you can install Visual Studio 2015 properly with Custom Installation options and make sure checked Visual C++ on Programming Language Section.
2. Git for Windows ([Link](https://git-scm.com/downloads))
3. MinGW ([Link](https://sourceforge.net/projects/mingw/files/))
4. Pkg-config ([Link](https://www.freedesktop.org/wiki/Software/pkg-config/))
5. Cmake ([Link](https://cmake.org/download/)) (For Linux, I suggested to install *cmake-curses-gui*; it has better interface)
6. Qt5 (for openCV GUI; [Link](http://doc.qt.io/qt-5/windows-support.html) -> follow the instructions to install Qt5)
7. CUDA ([link](https://developer.nvidia.com/cuda-downloads)) and cuDNN (you have to login first with Nvidia account to get cuDNN libraries. [Link](https://developer.nvidia.com/cudnn))
8. OpenCV. In this tutorial, I use 3.30 and Contrib included (I need some library from Contrib version). You can get it from git or [openCV](https://github.com/opencv/opencv/releases) and [openCV Contrib](https://github.com/opencv/opencv_contrib/releases)

Other options:

1. Threading block ([Link](https://www.threadingbuildingblocks.org/)) Intel® Only. NOTE: link already on the Building section
2. Intel® Integrated Performance Primitives ([Link](https://software.intel.com/en-us/intel-ipp)) NOTE: link already on the Building section
3. Caffe (I add caffe because of some reason. It’s just an options)
4. Java, if you want to use openCV with it. I use JDK instead JRE, it works. The solution will build the \*.jar files
5. Doxygen, if you want to build a Documentation

**Building the library**

1. Install Anaconda first. After done, open command prompt and change it to python 3.5

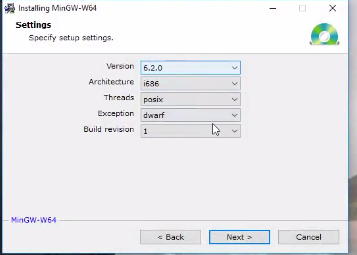


\*Note: ignores the version. I found it on Google. ([Source](https://sassoftware.github.io/sas_kernel/install.html))

|  |
| --- |
| *> conda update conda*  *> conda install python=3.5 -y* |

It takes a minute to done.

1. Install MinGW.

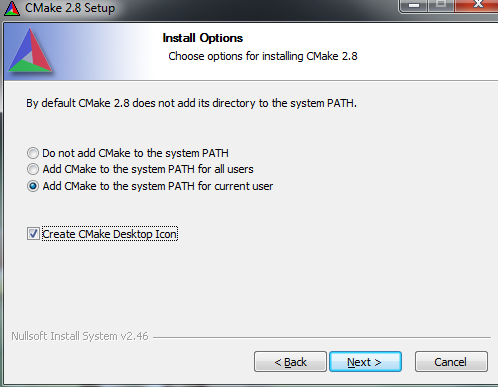


\*Note: ignores the version. I found it on Youtube. ([Source](https://www.youtube.com/watch?v=f3Ion00p78M))

First, change your architecture if you use 64-bit version.

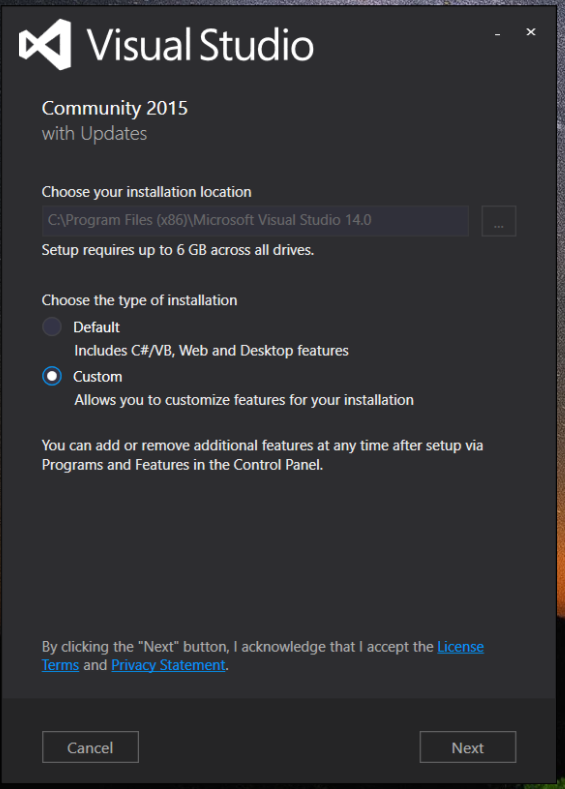
With MinGW, you already installed BLAS, GCC etc include msys (you can use Linux command on Windows with this)

1. Install pkg-config, cmake, git. Most of them easily to install.



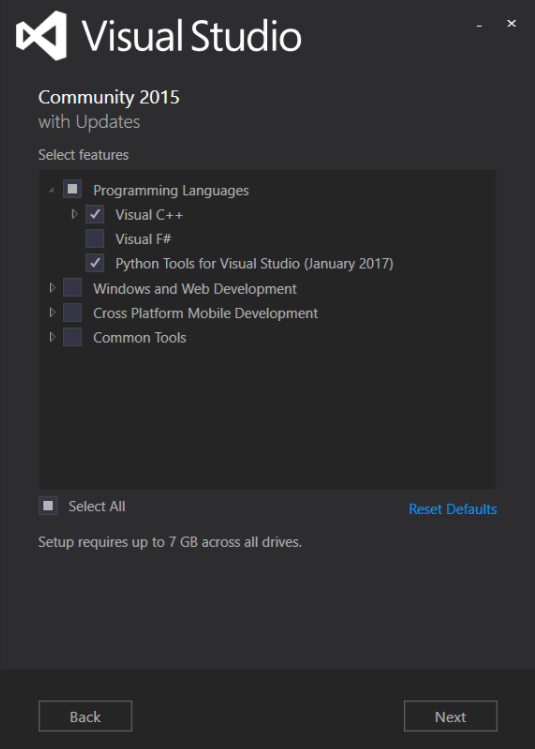
\*Note: ignores the version. I use 3.9.1 version. I found it on Google. ([Source](http://tulip.labri.fr/TulipDrupal/?q=node/1081))

1. Install Visual C++ Redistribute Packages. Its easy but you have to restart your computer, so make user save all of your things (“*Everything not save will be lost*” – Nintendo “Quit Screen” message). Then install Visual Studio 2015



\*Note: of course! Here’s the [source](http://www.learnopencv.com/install-opencv3-on-windows/). \*Captain flies away\*

Don’t forget to choose this options:



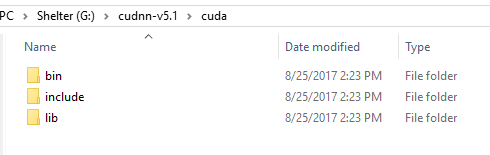
\*Note: Same source.

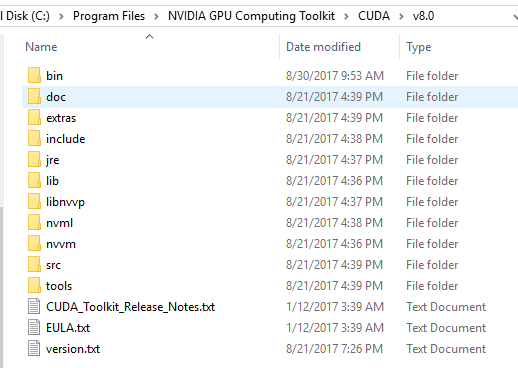
1. To install Qt5, please refers to the instruction that included from the link.
2. Install CUDA and cuDNN. I’m sorry in this tutorial I don’t include openCL configuration. I have not an experience with AMD GPU. Choose custom build to make sure all of component will be installed.



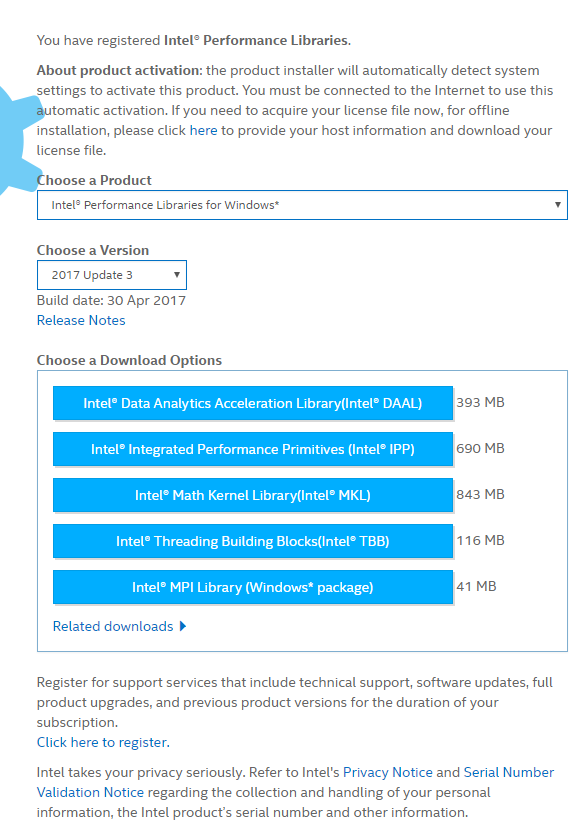
\*Note: you got me. I use **CUDA 8**. ([Source](http://embedonix.com/articles/machine-learning/compiling-caffe-with-cuda-and-cudnn-support-on-windows-from-source/))

To install cuDNN, just extract the cuDNN files, there’s a several folder which matches with CUDA folder. Put files from cuDNN bin to CUDA bin, include to include and so on.





1. Install java if you want it.
2. In this tutorial, I use Math Kernel Library for Intel® -based system ([Link](https://software.intel.com/en-us/mkl)) because it already include BLAS, LAPACK and many others for C and FORTRAN and TBB. Before you download it, you have to make an account. Choose for Windows

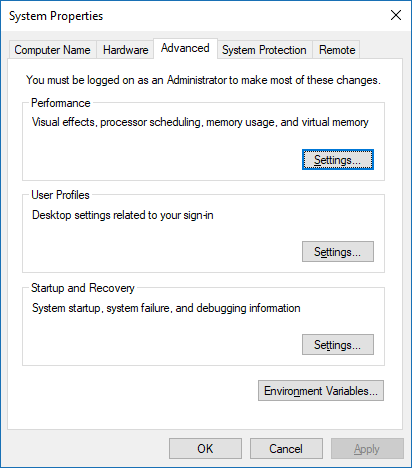


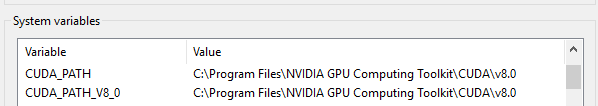
Choose Windows

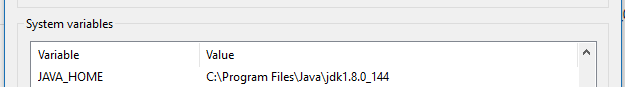
\*Note: No, I got it myself

You can downloat all of it for another purposes. In this tutorial we just use TBB and MKL.

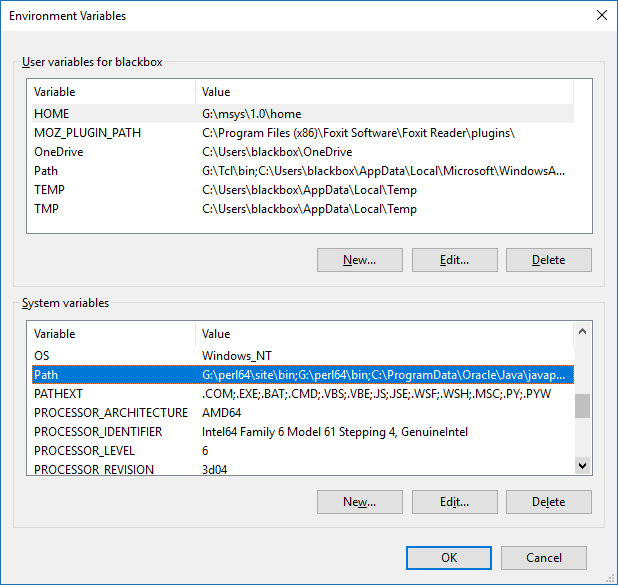
1. Make sure you had install all of dependecies (or I’ll make another User Manual to install the dependecies) in order correctly and add it to your PATH. Check it on System Properties.





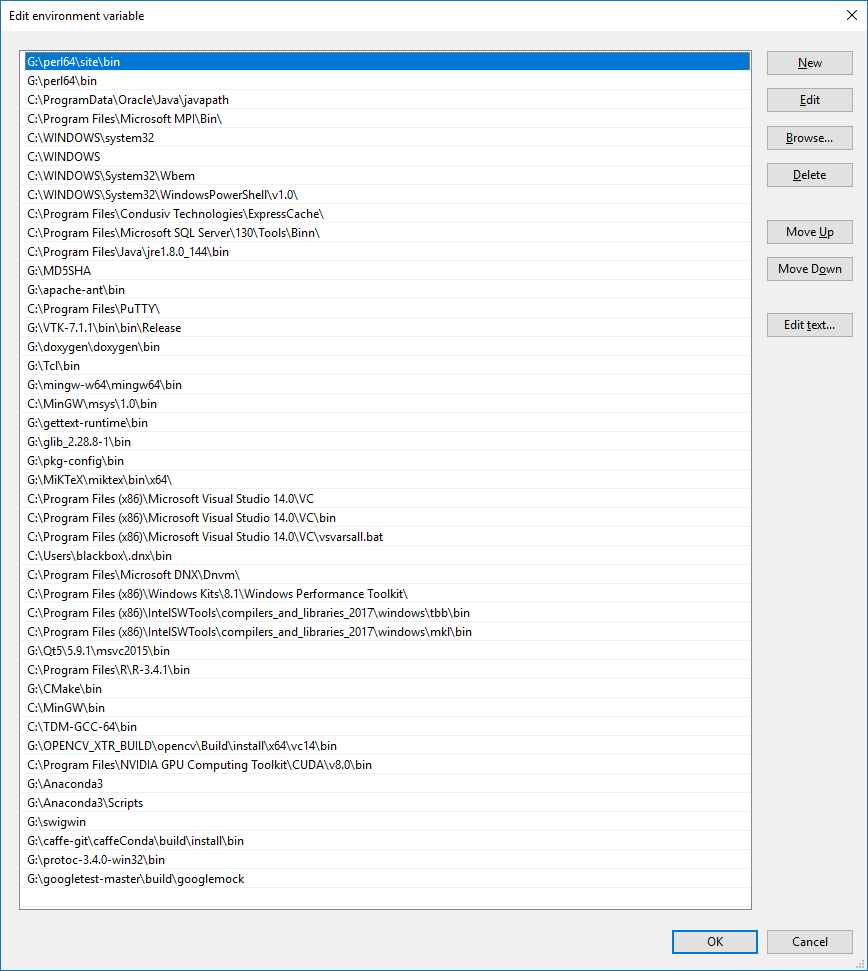


PATH



PATH (1)

Edit (2)



CUDA

Caffe

Anaconda

CMake

Qt5

MKL

TBB

Visual Studio 2015

Pkg-config

MinGW

doxygen

\*Note: That’s mine.

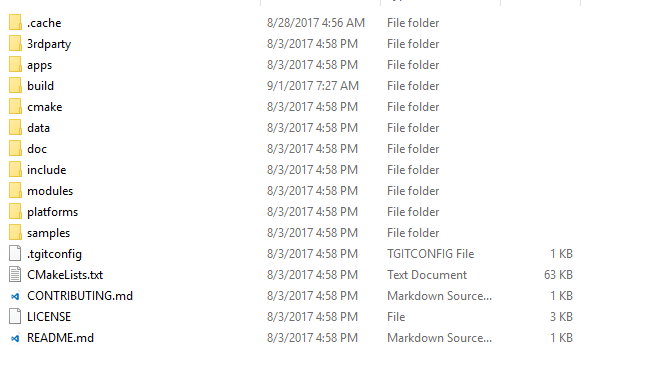
1. Download opencv and opencv contrib from the link I give or with git

|  |
| --- |
| *> git clone https://github.com/opencv/opencv.git*  *> git clone https://github.com/opencv/opencv\_contrib.git* |

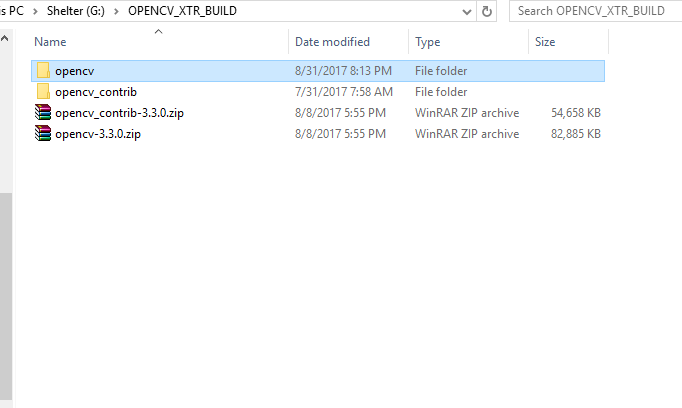
Now we are ready to install it

**Installing OpenCV**

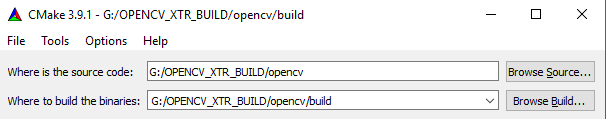
1. Open opencv directory. Make a new folder named “build”



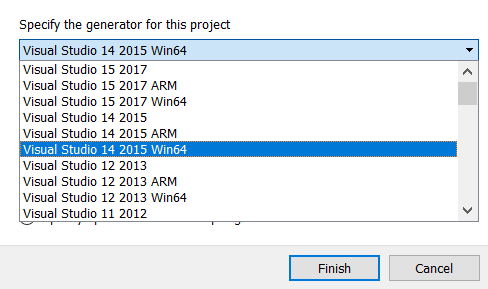
And I suggested you to put opencv and opencv-contrib in one folder



1. Open Cmake. Put your opencv folder as a source code and opencv/build to build the binaries



Then click Configure. Choose the generator. I use Visual Studio 14 2015 Win64 as my generator.

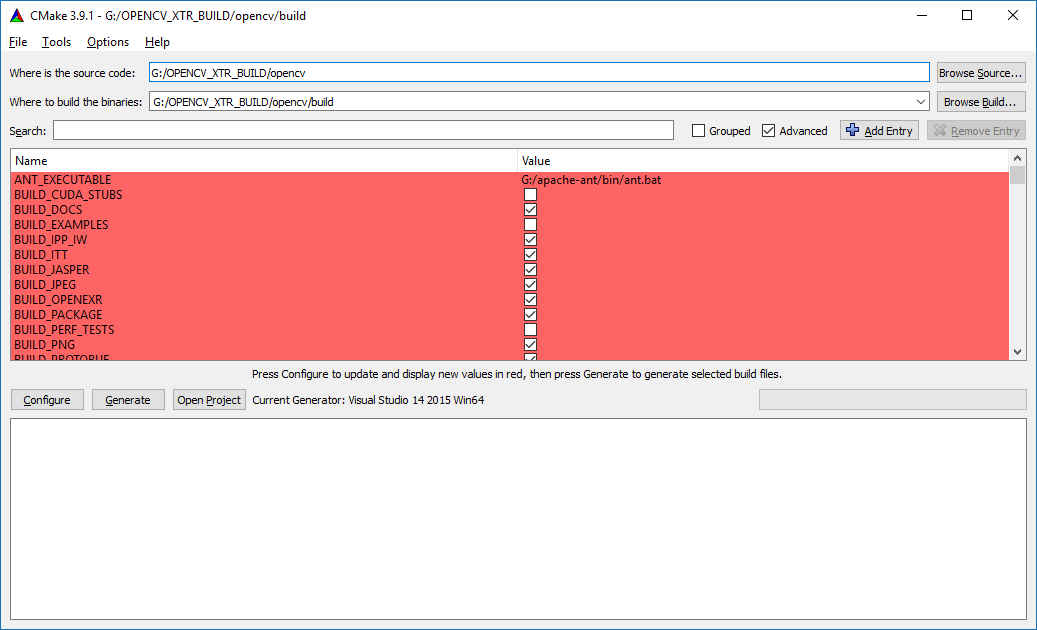


Click finish

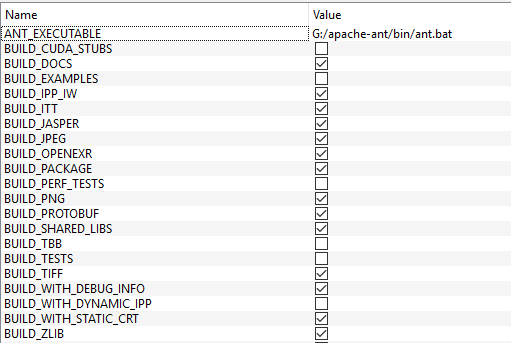
1. Next step, you will find a red line in cmake like this. Checked the advanced options. In this step you may not see an options like mine, you have to check this first:

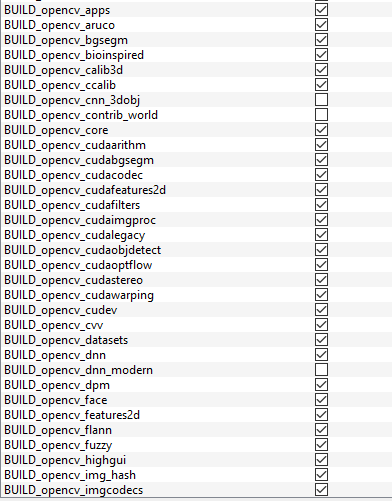
* WITH\_QT
* WITH\_OPENGL
* OPENCV\_EXTRA\_MODULES\_PATH -> fill with opencv-contrib/modules

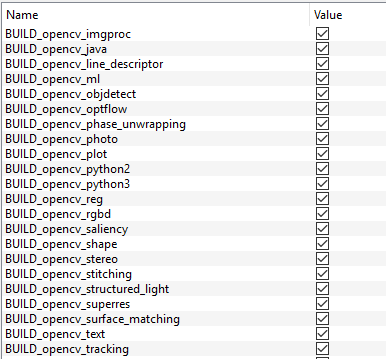
And if the python options its not present, you must fill it too. Find them with Search options

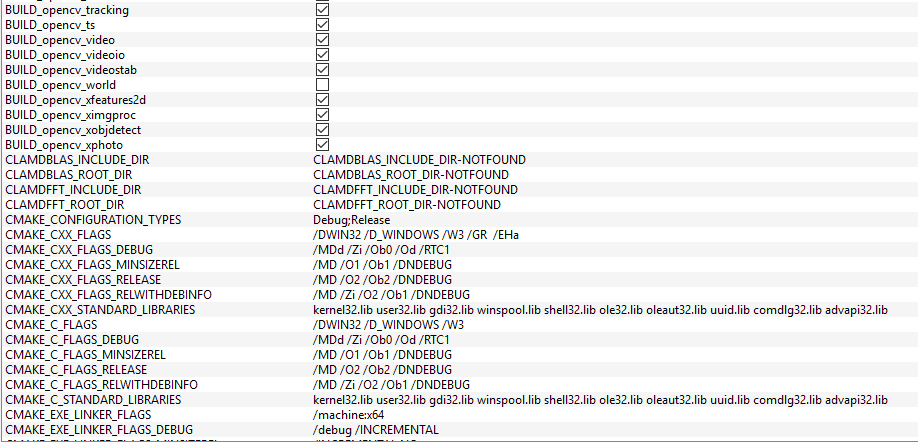


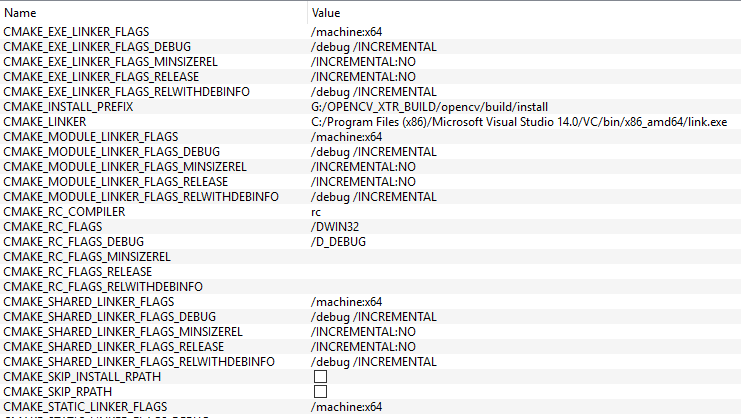
Now you have to configure some of options. This is my configuration

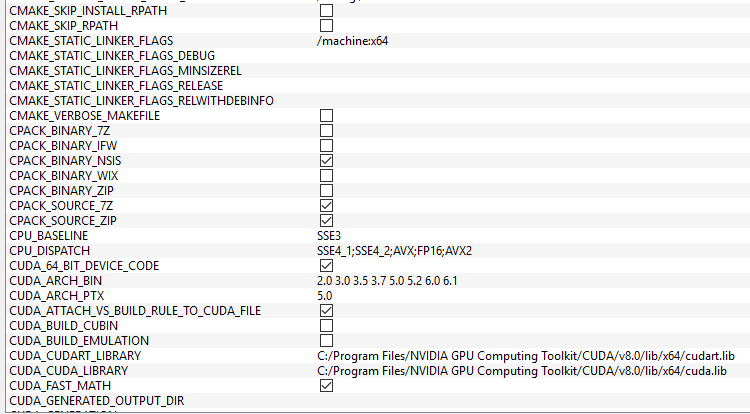


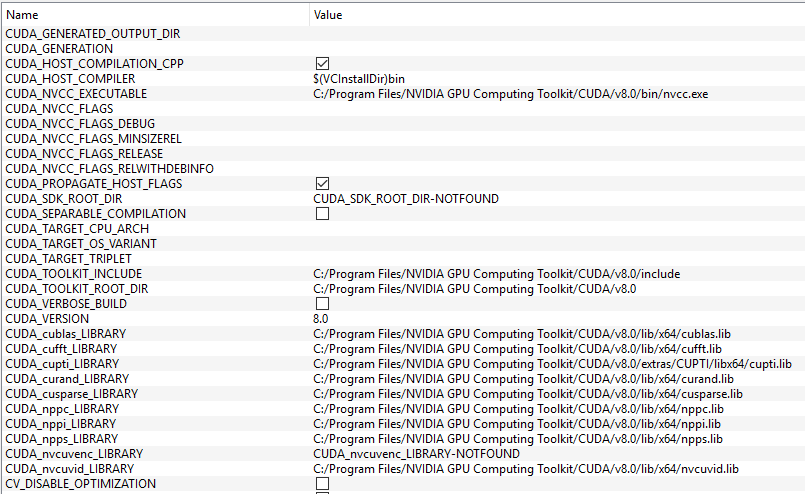


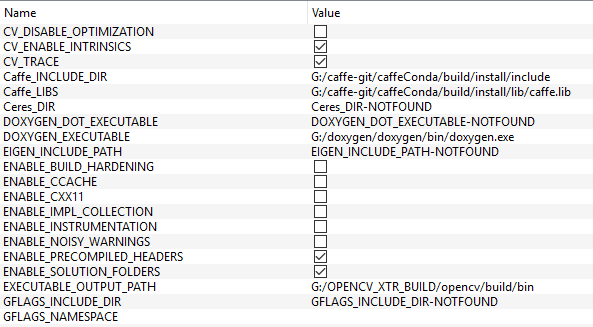




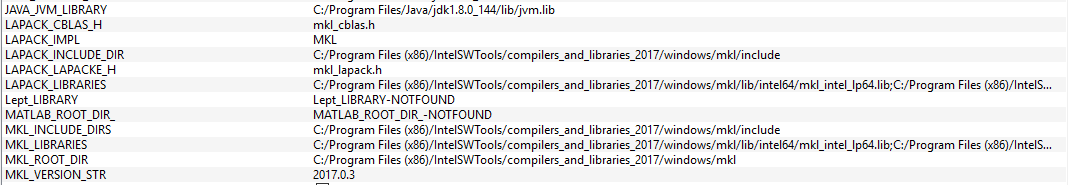


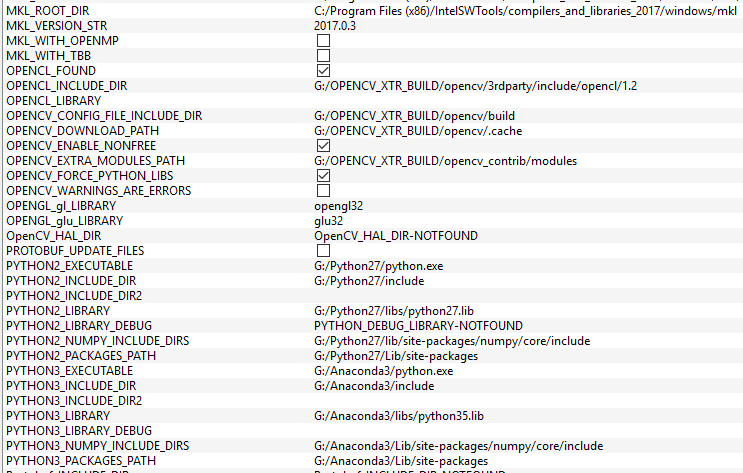


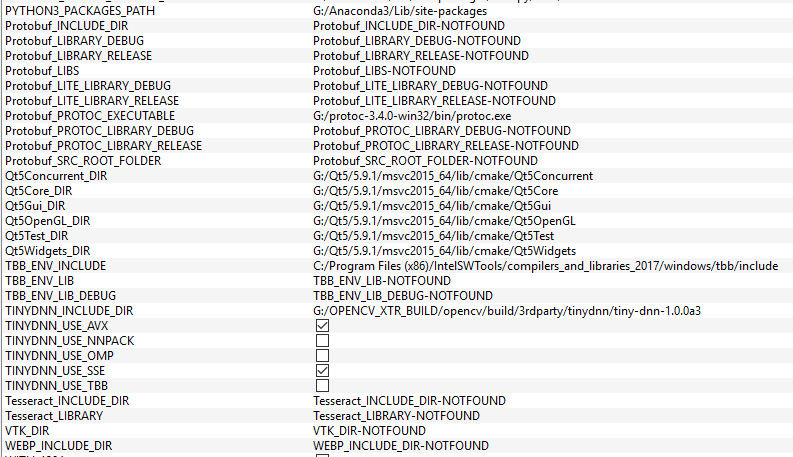


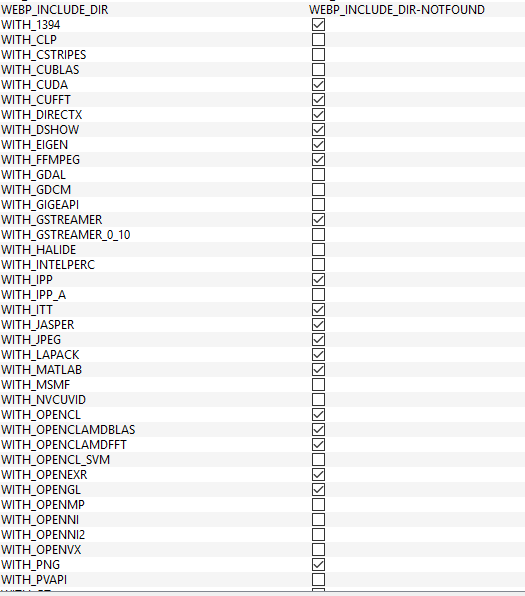


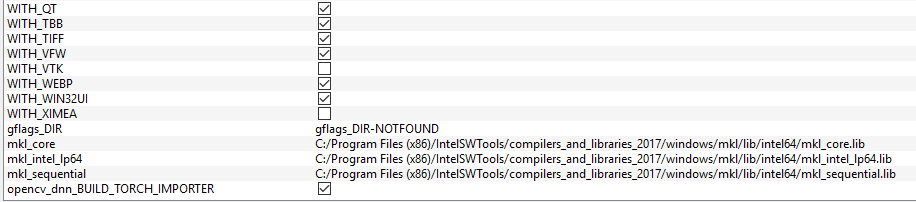






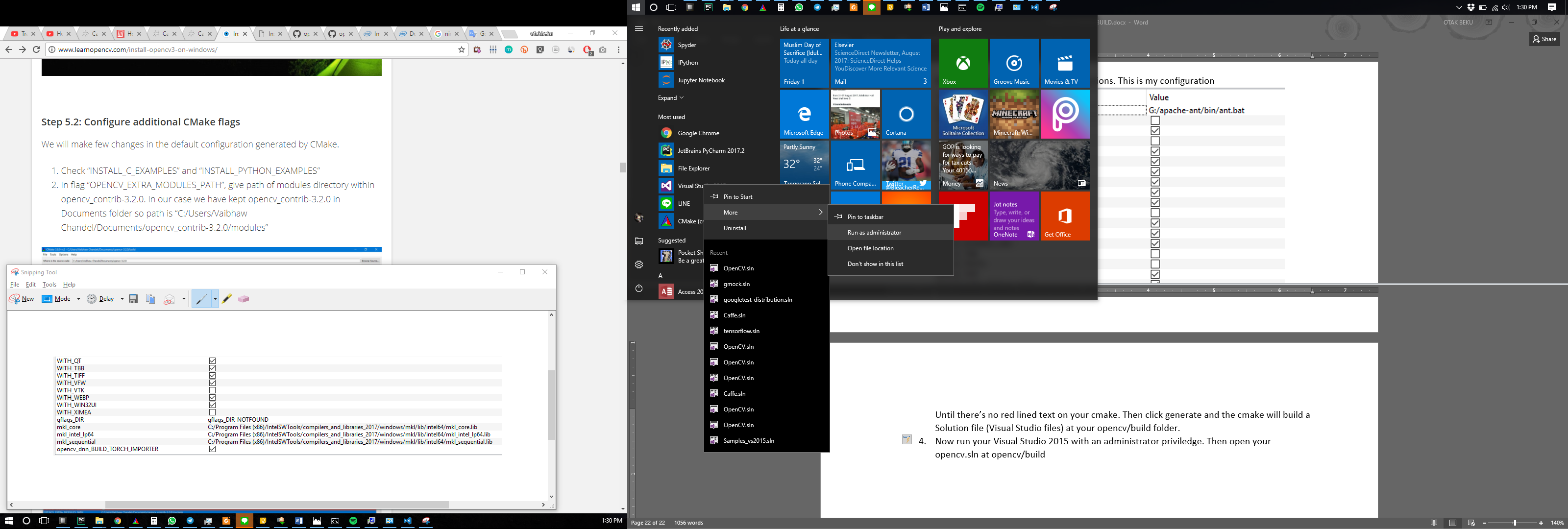


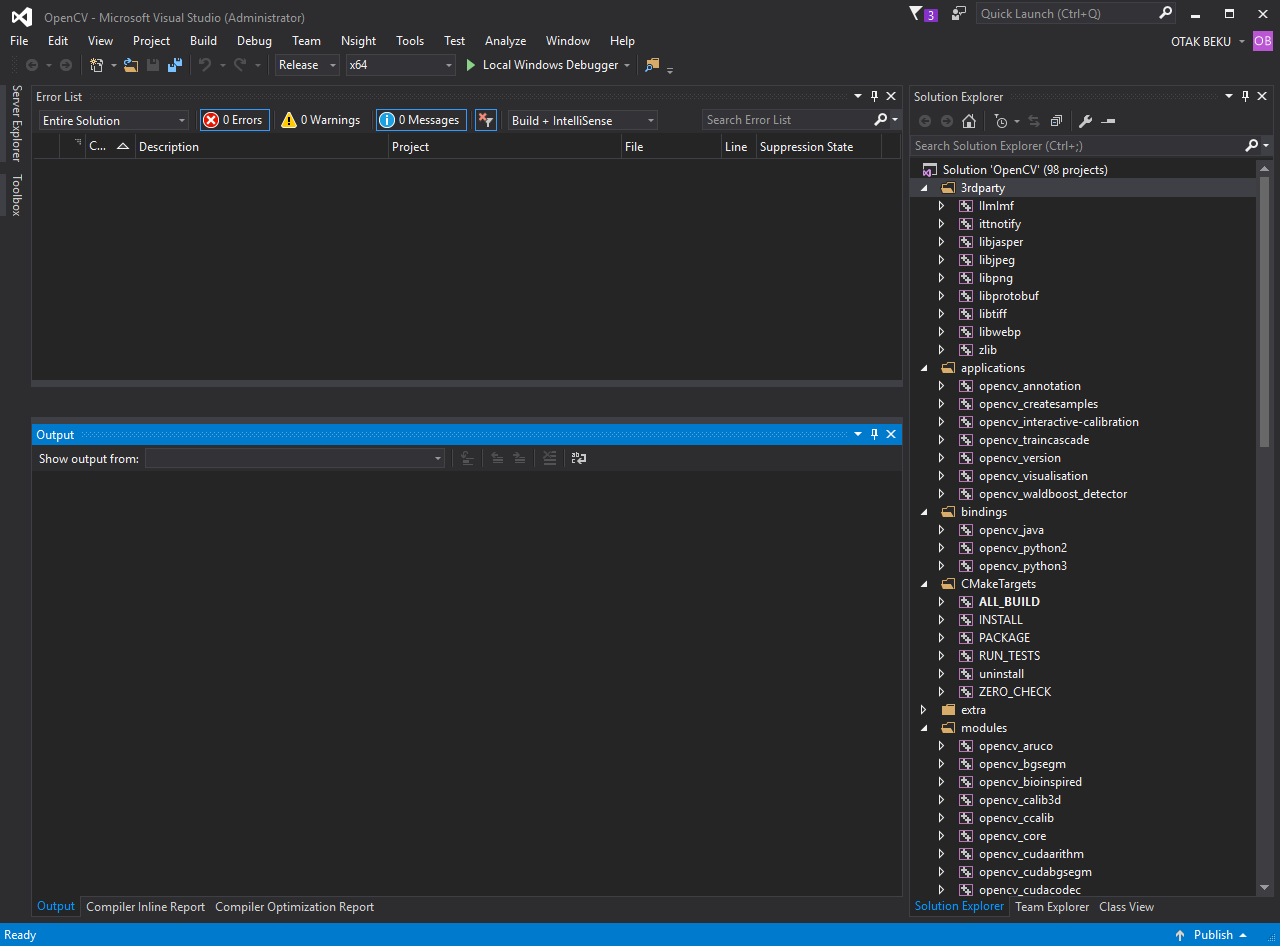




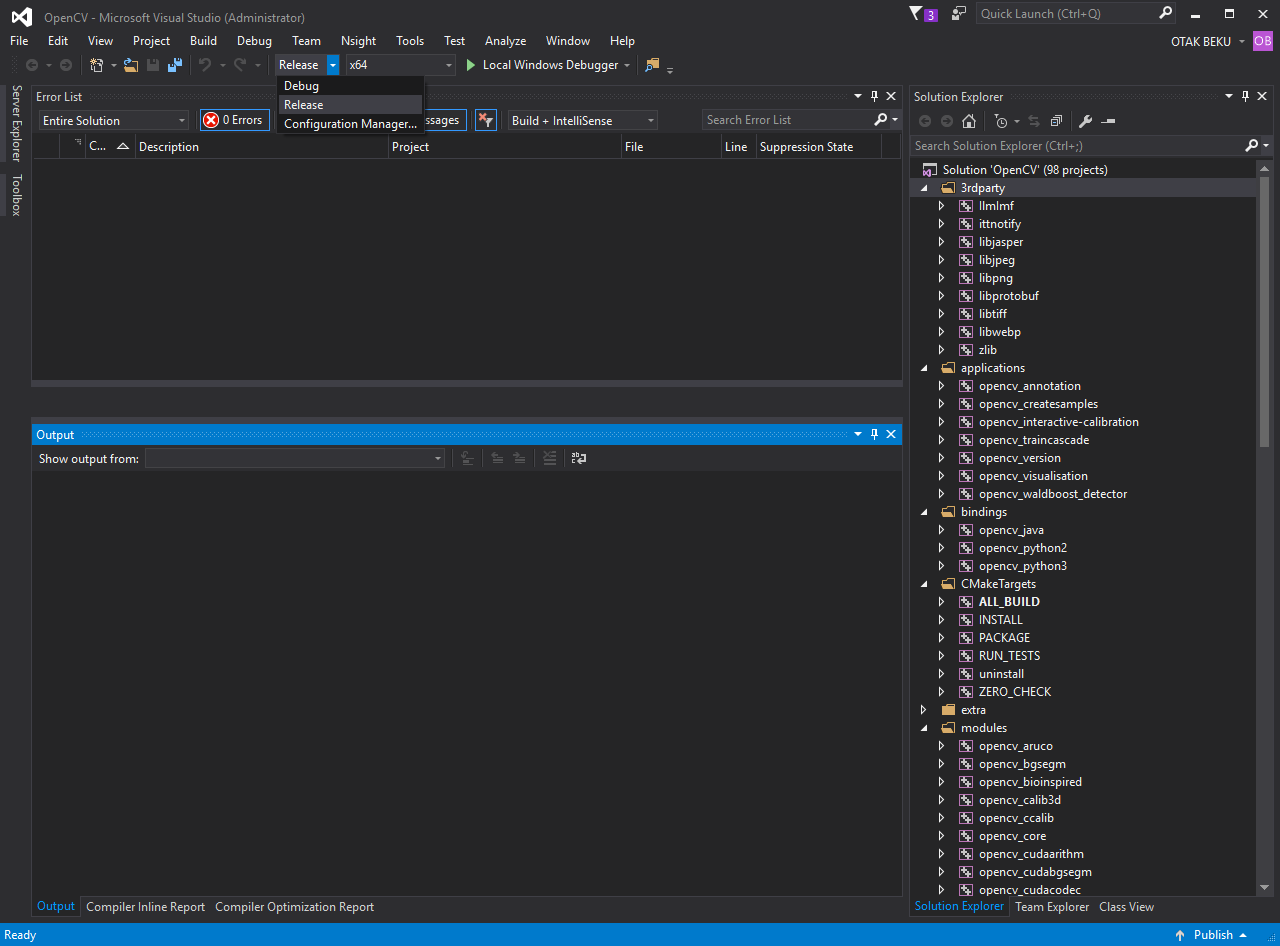
Don’t worry about some of checked options at the beginning. If the dependecies had not been installed, then cmake will not configure it and ignores it. But some of cases, you will get and error. Don’t panic, just read what you should be unchecked, then configure it again. Until there’s no red lined text on your cmake. Then click generate and the cmake will build a Solution file (Visual Studio files) at your opencv/build folder.

1. Now run your Visual Studio 2015 with an administrator priviledge. Then open your opencv.sln at opencv/build

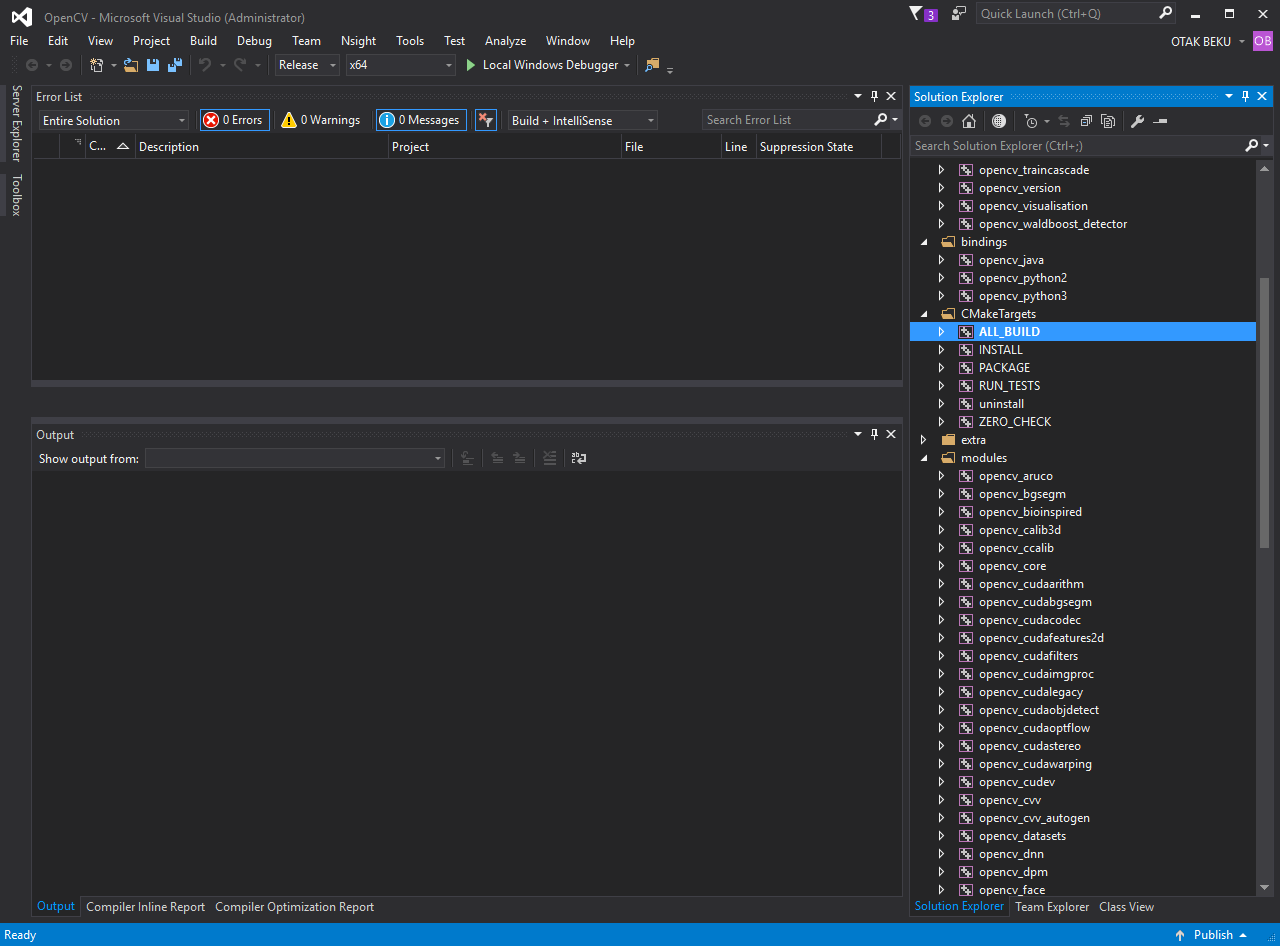




Note for this. I got problem with debug but in release options, there’s no problem. So, if you have no problem with debug configuration, you can keep going. But if you wan to change to release, here

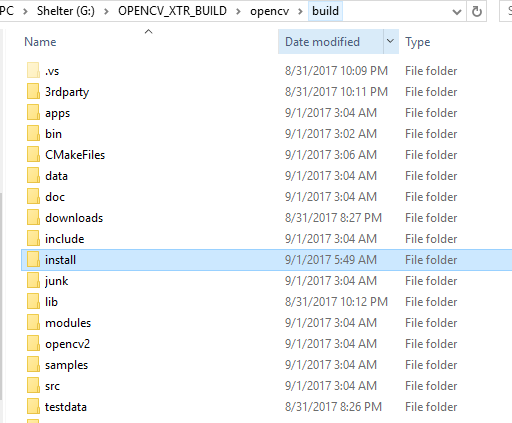


Now right-click on ALL\_BUILD in CmakeTargets and choose build. This will take a long time. With my specifications, I got 5 hour until its done.

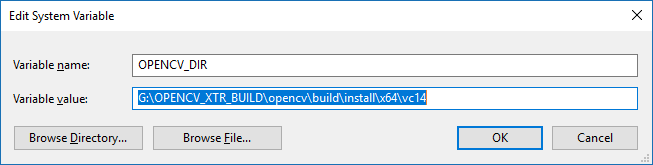


When you have done with ALL\_BUILD, now do it again with INSTALL. It takes less longer.

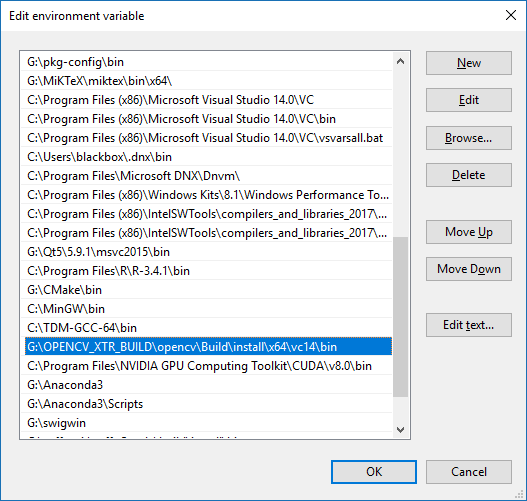
After that, you will got folder named “install” in opencv/build.



In install folder, there’s VC14 folder. Put it on your environment. Here’s an example

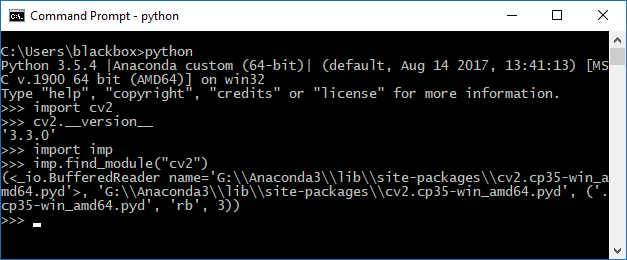


And put /VC14/bin to your PATH



Now its done!

Try it with python on cmd



Import imp to see where’s you cv2.pyd files was stored. Usually in site-packages unless you choose different path.

I hope this tutorial help you. If you had any problem or anything else, you can ask [me](mailto:iceramngl@gmail.com) or you can find it first with Google. Thank you

Here’s a potato

